PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-267886
(43)Date of publication of application : 29.09.2000

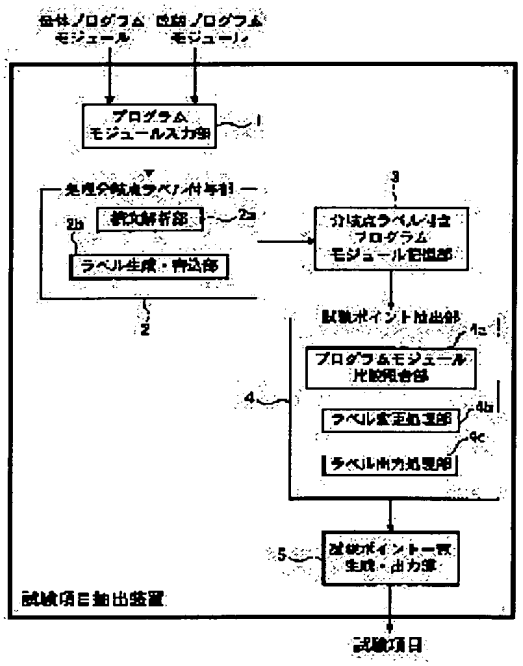
(51)Int.Cl. G06F 11/28

(21)Application number : 11-076704
(22)Date of filing : 19.03.1999

(71)Applicant : NTT COMMUNICATIONWARE CORP
(72)Inventor : NAGASHIMA AKIRA
MATSUTANI TADAYUKI
HATTORI NOBUTAKA
ANDO KINGO

(54) METHOD AND DEVICE FOR EXTRACTING SOFTWARE TEST ITEM AND RECORDING MEDIUM

(57)Abstract:
PROBLEM TO BE SOLVED: To automatically extract test items in just proportion when a software test is performed.
SOLUTION: A processing branch point label attaching part 2 respectively separately attaches labels to a matrix program module and a revised program module inputted from a program module inputting part 1 and stores them in a branch point labeled program module storing part 3. Then, the stored program modules are read, a test point extracting part 4 extracts test points, and test items are outputted from a test point list generating and outputting part 5 on the basis of the extracted results.



LEGAL STATUS

[Date of request for examination] 23.03.2001
[Date of sending the examiner's decision of rejection] 30.03.2004
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

JP2000-267886A

[0019] 2. Operation of this embodiment

The processing operation of the test item extraction apparatus according to this embodiment will be now described in an order of summary of operation and then an actual operation.

2.1. Summary of operation

To extract all processing routes of a program as test items, test routes covering all processing branch points should be considered. Considering a matter in that completeness of processing routes of the program is almost equal to the completeness of branch points of the program, the test execution points are extracted. In a process to extract test items for a program module updated by addition of functions, a technique to extract all modified/added branch points is established and processing routes split from the branch points are taken to as test target routes.

[0020]

As the technique to extract the test target routes, a program module before addition/modification and a program module (updated module) after the addition/modification are compared and differential information is analyzed, and when a branch route is added or when the processing contents of a branch route is modified or branch conditions are changed, a new label is given to the branch route in the updated module, and the label is output. As a result, the test routes for the updated module are extracted without fail.

[0021] 2.2 Actual operation

Fig. 3 is a flowchart of operation for extracting test points from an updated program module. When an original program module and an updated program module are entered with the program module input unit 1 (S1), the processing branch point labeling unit 2 executes a process to insert labels to these two modules independently (S2, S3).

[0022]

Fig. 4(a) shows a format of a label. The label to be inserted at a branch clause of the program modules comprises a class name L1, a member function name (method name) L2, a branch sentence type L3 such as an "if, for, switch" sentence, an in-function running number (or label number) L4 that is an index of a focused function, and a clause type L5 such as a "then, else, case" clause. Based on the label information, the position of a branch sentence, or the position (processing route) given the label, in the program, is specified. That is, as the class name and the member function name, the names used for the focused function are used as they are. Therefore, because of this format, only from a label, it can be recognized "which class, which member function, which branch point, and which branch clause the label is attached to".

[0023]

Fig. 4(b) shows an example in which labels are inserted at branch clauses B1 and B2 split from a branch point P in a program module. In

this figure, lines having a "printf" sentence shown by ① and ② are labels. By inserting such a label in the program module, in a case of ①, information indicating that "this is a then-clause process with a running number of 1000 in an if-sentence in a procl member function belonging to a Proc class" is output in the form of the printf sentence when a debugger is driven to execute a single test.

[0026]

Fig. 6 is a flowchart of a process to compare the processing contents of these branch clauses. At first, the processing contents of the branch clause of the updated source module with the processing contents of the corresponding branch clause of the original source module are compared and their differences are analyzed (S30). Then, it is determined whether the branch clauses have different labels (S31). When the labels are identical (S31; NO), then it can be determined whether the processing contents of the branch clauses are different (S32). When the processing contents are different (S32; YES), it can be recognized that the branch conditions are the same but the processing contents of the branch clauses are modified. In this case, the label of the updated program module is changed and the new label is output to make a notice of a test route (S33). When the processing contents are the same (S32; NO), on the other hand, it can be recognized that the focused branch clauses before and after the modification are the same, and in this case, this process of Fig. 6 ends.

[0027]

When it is determined at step S31 that the labels are different (S31; YES), then it is determined whether the processing contents of the branch clauses are different (S34), in the same way of step S32. When the processing contents are the same (S34; NO), it can be recognized that the processing contents are unchanged but only label numbers are different. In this case, the label number of the branch clause of the updated program module is changed to the label number given to the corresponding branch clause of the original program module (S35). When the processing contents of the branch clauses are different (S34; YES), that is, when the labels and the processing contents of the branch clauses are both different, it can be recognized that the updated program module has the modified processing contents. In this case, the label of the branch clause of the updated program module is output to make a notice of a test route (S36). As described above, a process to compare the processing contents of branch clauses is executed.

[0028]

Back to Fig. 3. After the process to compare the contents of branch clauses is completed at step S7 as described above, it can be determined whether the comparison process has been done for all branch clauses of the updated program module (S8). When the comparison process has been done for the all branch clauses (S8; YES), the test item extraction process is completed. When the comparison process has not been done for the all

branch clauses (S8; NO), a pointer is moved to a next branch clause of the updated program module (S9) and the process after step S5 is performed.
[0029]

As described above, the test point extraction unit 4 sequentially outputs and gives labels representing test target routes to the test point list creation/output unit 5, which creates a file by collecting the label group indicating all the test routes in an appropriate form and provides it as a test item list to a test manager.

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（Ａ）　(11)特許出願公開番号
特開2000－267886
（P2000－267886A）
(43)公開日　平成12年９月29日(2000.9.29)

(51)Int.Cl.'	識別記号	ＦＩ	ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ　11/28	３４０	Ｇ０６Ｆ　11/28	３４０Ａ　５Ｂ０４２

審査請求　未請求　請求項の数８　ＯＬ　（全８頁）

(21)出願番号　特願平11－76704

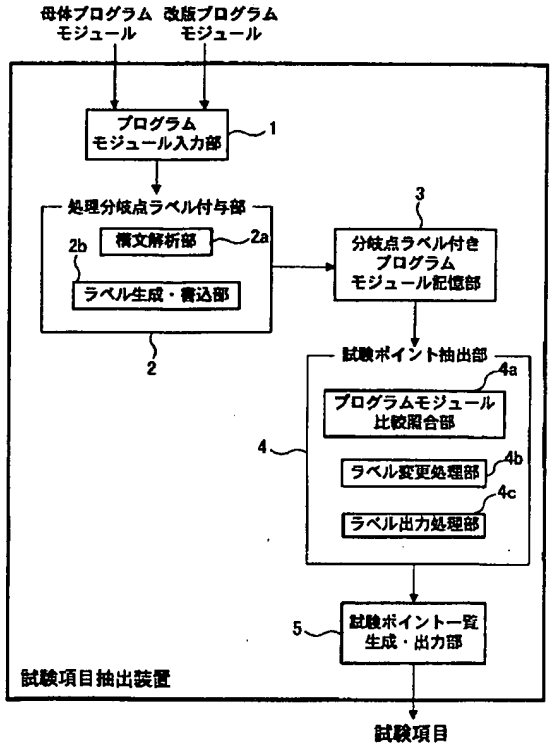
(22)出願日　平成11年３月19日(1999.3.19)

(71)出願人　397065480
エヌ・ティ・ティ・コミュニケーションウェア株式会社
東京都港区港南一丁目９番１号
(72)発明者　長島　彰
東京都港区港南一丁目９番１号　エヌ・ティ・ティ・コミュニケーションウェア株式会社内
(74)代理人　100098084
弁理士　川▲崎▼　研二

最終頁に続く

(54)【発明の名称】　ソフトウェア試験項目抽出方法、ソフトウェア試験項目抽出装置および記録媒体

(57)【要約】
【課題】　ソフトウェア試験を実施する際、試験項目を過不足なく自動的に抽出することができるようにする。
【解決手段】　プログラムモジュール入力部１から入力された母体プログラムモジュールおよび改版プログラムモジュールに対して、処理分岐点ラベル付与部２においてそれぞれ別個にラベルを付与し、分岐点ラベル付プログラムモジュール記憶部３に記憶する。そして、記憶されたこれらのプログラムモジュールを読み出して試験ポイント抽出部４で試験ポイントを抽出し、この抽出結果に基づいて試験項目を試験ポイント一覧生成・出力部５から出力する。



1

【特許請求の範囲】

【請求項１】　ソフトウェアの試験を行う際に、当該試験の対象となる項目を抽出するソフトウェア試験項目抽出方法において、

プログラムモジュールの構文を解析して処理の分岐点を検索する過程と、

検索された処理の分岐点から派生する処理ルートに対して、該処理ルートを特定する識別情報を付与する過程とを有することを特徴とするソフトウェア試験項目抽出方法。

【請求項２】　ソフトウェアの試験を行う際に、当該試験の対象となる項目を抽出するソフトウェア試験項目抽出方法において、

第１のプログラムモジュール、および前記第１のプログラムモジュールに変更を加えた第２のプログラムモジュールの各々に対して、各モジュールの構文を解析して処理の分岐点を検索し、該分岐点から派生する処理ルート毎に該処理ルートを特定する識別情報を付与する識別情報付与過程と、

前記識別情報が付与された前記第１のプログラムモジュールと、前記識別情報が付与された前記第２のプログラムモジュールとを比較照合した結果に基づいて、前記第２のプログラムモジュールに付与された識別情報の中から適宜な識別情報を抽出し、抽出された識別情報によって特定される前記第２のプログラムモジュール内に記述された処理ルートに対する試験を試験項目とする試験項目抽出過程とを有することを特徴とするソフトウェア試験項目抽出方法。

【請求項３】　請求項２に記載のソフトウェア試験項目抽出方法において、

前記試験項目抽出過程では、前記第１のプログラムモジュールにおいて処理の分岐点に記述された分岐条件と、前記第２のプログラムモジュールにおいて該分岐点に相当する部分に記述された分岐条件とが異なる場合に、前記第２のプログラムモジュールにおける当該分岐点から派生する処理ルートに対して新たな識別情報を付与し、この識別情報を抽出することを特徴とするソフトウェア試験項目抽出方法。

【請求項４】　請求項２または３に記載のソフトウェア試験項目抽出方法において、

前記試験項目抽出過程では、前記第１のプログラムモジュールにおいて処理の分岐点から派生する処理ルートにおける処理の内容と、前記第２のプログラムモジュールにおいて該分岐点に相当する部分から派生する処理ルートにおける処理の内容とが異なる場合に、前記第２のプログラムモジュールにおける当該処理ルートに対して新たな識別情報を付与し、この識別情報を抽出することを特徴とするソフトウェア試験項目抽出方法。

【請求項５】　請求項１に記載のソフトウェア試験項目抽出方法において、

2

前記プログラムモジュールは、オブジェクト指向言語によって記述され、

前記識別情報には、クラス情報と、メンバ関数情報と、該メンバ関数内において前記処理ルートを特定する情報とが含まれることを特徴とするソフトウェア試験項目抽出方法。

【請求項６】　請求項２～４のいずれかに記載のソフトウェア試験項目抽出方法において、

前記第１のプログラムモジュールおよび前記第２のプログラムモジュールは、ともにオブジェクト指向言語によって記述され、

前記識別情報には、クラス情報と、メンバ関数情報と、該メンバ関数内において前記処理ルートを特定する情報とが含まれることを特徴とするソフトウェア試験項目抽出方法。

【請求項７】　請求項１～６のいずれかに記載のソフトウェア試験項目抽出方法を実行することを特徴とするソフトウェア試験項目抽出装置。

【請求項８】　請求項１～６のいずれかに記載のソフトウェア試験項目抽出方法を実行するプログラムを記録した記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、ソフトウェア試験において、試験項目を抽出する際に用いて好適なソフトウェア試験項目抽出方法、ソフトウェア試験項目抽出装置および記録媒体に関する。

【０００２】

【従来の技術】ソフトウェアの開発工程では、設計、製造工程の後に単体、結合、総合の各試験が行われ、これらの試験を通じてソフトウェアの品質特性が機能単位に確認される。特に、単体試験は製造直後の試験であることから、その担う役割は大きい。この単体試験では、個々のプログラムモジュール（サブルーチン、手続き、関数など）単位に試験を実施する必要があるため、試験項目数が多く、それに伴い試験工数も多くを要する。

【０００３】この試験項目を抽出する際、次のようなプロセスで試験担当者（プログラマ）が目視で確認しながら試験対象とすべきルートを決定していた。図７は、単体プログラム内の処理分岐モデルを用いて、この試験項目抽出方法のプロセスを示したものである。同図に示すように、試験項目Ａに対する試験ルート（ある条件で動作するプログラムの道筋）Ｒ１、試験項目Ｂに対する試験ルートＲ２というようにそれぞれの試験ルートを想定し、試験項目Ａの内容として、「試験ルート：処理分岐点ａ→ｅ，出力：Output1」、試験項目Ｂの内容として、「試験ルート：処理分岐点ａ→ｂ→ｄ，出力：Output4」といった処理ルートおよび分岐条件を自然言語で表記していた。また、母体となるプログラムモジュールに対して、機能追加・変更等の処理を盛り込むことで改

版されたプログラムモジュールに対する単体試験を実施する場合、試験担当者は、この機能追加等に伴って変更された分岐処理部分を意識して、上述のような自然言語で処理ルートを表現し、試験項目としていた。

【０００４】

【発明が解決しようとする課題】このように、試験担当者が目視によって試験項目を抽出していたため、個人のスキルによる部分が大きく、試験項目の抽出漏れを避けることができなかった。また、各試験項目は上述のように曖昧な自然言語で表現されていたため、膨大なプログラムモジュールから直ちに試験ルートを特定することは困難であった。さらに、改版モジュールに対する試験では、上述のような問題に加えて、母体となるモジュールに対する試験が終了している部分まで、重複して試験項目としてしまうことがあった。つまり、ソフトウェア単体試験を実施する際、試験項目の抽出段階において、抽出漏れや冗長といった要素を排除することができず、迅速かつ信頼性の高い試験を実施することができなかったり、無駄な試験が行われる要因ともなっていた。

【０００５】そこで、この発明は、このような課題に着目してなされたもので、ソフトウェア試験を実施する際、試験項目を過不足なく自動的に抽出することができるソフトウェア試験項目抽出方法、ソフトウェア試験項目抽出装置および記録媒体を提供することを目的とするものである。

【０００６】

【課題を解決するための手段】上記目的を達成するために、第１発明のソフトウェア試験項目抽出方法は、ソフトウェアの試験を行う際に、当該試験の対象となる項目を抽出するソフトウェア試験項目抽出方法において、プログラムモジュールの構文を解析して処理の分岐点を検索する過程と、検索された処理の分岐点から派生する処理ルートに対して、該処理ルートを特定する識別情報を付与する過程とを有することを特徴とするものである。

【０００７】第２発明のソフトウェア試験項目抽出方法は、ソフトウェアの試験を行う際に、当該試験の対象となる項目を抽出するソフトウェア試験項目抽出方法において、第１のプログラムモジュール、および前記第１のプログラムモジュールに変更を加えた第２のプログラムモジュールの各々に対して、各モジュールの構文を解析して処理の分岐点を検索し、該分岐点から派生する処理ルート毎に該処理ルートを特定する識別情報を付与する識別情報付与過程と、前記識別情報が付与された前記第１のプログラムモジュールと、前記識別情報が付与された前記第２のプログラムモジュールとを比較照合した結果に基づいて、前記第２のプログラムモジュールに付与された識別情報の中から適宜な識別情報を抽出し、抽出された識別情報によって特定される前記第２のプログラムモジュール内に記述された処理ルートに対する試験を試験項目とする試験項目抽出過程とを有することを特徴とするものである。

【０００８】第３発明のソフトウェア試験項目抽出方法は、上記第２発明のソフトウェア試験項目抽出方法において、前記試験項目抽出過程では、前記第１のプログラムモジュールにおいて処理の分岐点に記述された分岐条件と、前記第２のプログラムモジュールにおいて該分岐点に相当する部分に記述された分岐条件とが異なる場合に、前記第２のプログラムモジュールにおける当該分岐点から派生する処理ルートに対して新たな識別情報を付与し、この識別情報を抽出することを特徴とするものである。

【０００９】第４発明のソフトウェア試験項目抽出方法は、上記第２または第３発明のソフトウェア試験項目抽出方法において、前記試験項目抽出過程では、前記第１のプログラムモジュールにおいて処理の分岐点から派生する処理ルートにおける処理の内容と、前記第２のプログラムモジュールにおいて該分岐点に相当する部分から派生する処理ルートにおける処理の内容とが異なる場合に、前記第２のプログラムモジュールにおける当該処理ルートに対して新たな識別情報を付与し、この識別情報を抽出することを特徴とするものである。

【００１０】第５発明のソフトウェア試験項目抽出方法は、上記第１発明のソフトウェア試験項目抽出方法において、前記プログラムモジュールは、オブジェクト指向言語によって記述され、前記識別情報には、クラス情報と、メンバ関数情報と、該メンバ関数内において前記処理ルートを特定する情報とが含まれることを特徴とするものである。

【００１１】第６発明のソフトウェア試験項目抽出方法は、上記第２～第４発明のいずれかのソフトウェア試験項目抽出方法において、前記第１のプログラムモジュールおよび前記第２のプログラムモジュールは、ともにオブジェクト指向言語によって記述され、前記識別情報には、クラス情報と、メンバ関数情報と、該メンバ関数内において前記処理ルートを特定する情報とが含まれることを特徴とするものである。

【００１２】第７発明のソフトウェア試験項目抽出装置は、上記第１～第６発明のいずれかのソフトウェア試験項目抽出方法を実行することを特徴とするものである。

第８発明の記録媒体は、上記第１～第６発明のいずれかのソフトウェア試験項目抽出方法を実行するプログラムを記録したものである。

【００１３】なお、このプログラムをＦＤ、ＣＤ－ＲＯＭ、テープメディア、ＤＶＤ－ＲＡＭ等の記録媒体に格納して頒布する（ネットワーク経由でこのプログラムを配信したり、ダウンロードする行為も含む）ことによっても本発明を実施することができる。

【００１４】

【発明の実施の形態】この発明の好ましい実施の形態について、以下、添付図面を参照しつつ詳細に説明する。

１．実施形態の構成

１．１．機能構成

図１は、本実施形態に係るソフトウェア単体試験において用いられる試験項目抽出装置を示している。この装置は、プログラムモジュール入力部１と、処理分岐点ラベル付与部２と、分岐点ラベル付きプログラムモジュール記憶部３と、試験ポイント抽出部４と、試験ポイント一覧生成・出力部５とを備えて構成される。

【００１５】プログラムモジュール入力部１からは、改版前のプログラムモジュールであって、単体試験による検証済みの母体プログラムモジュール、およびこの母体プログラムモジュールを編集して機能追加作業が行われた改版プログラムモジュールとが入力される。なお、これらのプログラムモジュールは、オブジェクト指向技術に基づく設計を行うべく、ともにプログラミング言語Ｃ＋＋で記述されている。

【００１６】処理分岐点ラベル付与部２は、プログラムモジュール入力部１から入力された上記２つのプログラムモジュールの供給を受け、それぞれ別個独立にラベルを付与する処理を行う。つまり、入力されたプログラムモジュールに記述された構文を解析することにより、処理の分岐点（例えば、後述する図４（ｂ）の符号Ｐで示す部分）を検索し、さらにその分岐点から派生するルート（分岐節；例えば、後述する図４（ｂ）の符号Ｂ１，Ｂ２で示す部分）をサーチする構文解析部２ａと、この分岐節毎に当該分岐節を特定するラベルを生成し、このラベルを該分岐節中に書き込むラベル生成・書込部２ｂとを有する。プログラムモジュール入力部１を介して入力された母体プログラムモジュールおよび改版プログラムモジュールは、構文解析部２ａ、ラベル生成・書込部２ｂにおける処理によって別個独立にラベルが付与される。この分岐ラベルが埋め込まれた母体プログラムモジュールおよび改版プログラムモジュールは、分岐点ラベル付プログラムモジュール記憶部３で記憶・保持される。

【００１７】試験ポイント抽出部４では、分岐点ラベル付プログラムモジュール記憶部３から、これら２つのラベル付きプログラムモジュールを読み出して比較照合を行い、単体試験を実施すべきポイントを抽出する。この試験ポイント抽出部４は、母体プログラムモジュールと改版プログラムモジュールとを比較照合するプログラムモジュール比較照合部４ａと、両モジュールの比較照合結果に基づいて改版モジュールに対するラベルの変更を行うラベル変更処理部４ｂと、試験ポイントとなる改版プログラムモジュールの分岐節中に記述されたラベルを出力するラベル出力処理部４ｃとを有する。試験ポイント一覧生成・出力部５では、ラベル出力処理部４ｃから出力されたラベルの内容を整理・統合して試験担当者に見やすい形で出力する。

【００１８】１．２．ハードウェア構成

図２は、本実施形態に係る試験項目抽出装置のハードウェア構成の概略を示す図である。同図に示すように、本装置は、装置全体を統括・制御するＣＰＵ１００と、処理分岐点ラベル付与部２、試験ポイント抽出部４および試験ポイント一覧生成・出力部５における処理を実行するプログラムが展開されるＲＡＭ１０１と、種々の制御用プログラムが記憶されたＲＯＭ１０２と、母体プログラムモジュールおよび改版プログラムモジュール、さらにこれらのプログラムモジュールにラベルが付与されたモジュールが記憶・保持されるＨＤＤ（ハードディスク装置）１０３と、外部端末からネットワーク経由で処理対象となるプログラムモジュールを受け入れるためのインタフェースである通信制御部１０４とを備えて構成され、これらがバス１０５で相互に接続されている。

【００１９】２．実施形態の動作

続いて、本実施形態に係る試験項目抽出装置における処理動作について、動作の概要、実際の動作の順に説明する。

２．１．動作の概要

プログラムのすべての処理ルートを試験項目として抽出するためには、処理分岐点のすべてを網羅した試験ルートを想定することが必要である。ここで、プログラムの処理ルートの網羅性は、プログラムの分岐ポイントの網羅性にほぼ等しい点に着目して、試験実施ポイントを抽出する。そして、機能追加等によって改版されたプログラムモジュールに対する試験項目を抽出する過程では、変更・追加となった分岐点をすべて抽出する仕組みを確立し、この分岐点から派生する処理ルートを試験対象ルートとすることとした。

【００２０】この試験対象ルートを抽出する仕組みは、追加変更前のプログラムモジュールと、追加変更後のプログラムモジュール（改版モジュール）とを比較し、その差分情報を解析することで、分岐ルート追加された場合や、分岐ルートは同じであっても分岐ルート内の処理内容が変更されたり分岐条件が変更された場合には、改版モジュールにおける当該分岐ルートに対して新たにラベルを付与し、このラベルを出力することで改版モジュールにおける試験ルートを過不足なく抽出するものである。

【００２１】２．２．実際の動作

図３は改版プログラムモジュールから試験ポイントを抽出する際の動作を示すフローチャートである。まず、母体プログラムモジュールおよび改版プログラムモジュールがプログラムモジュール入力部１に入力されると（Ｓ１）、処理分岐点ラベル付与部２において、これら２つのモジュールに対して別個独立にラベルを書き込む処理が実行される（Ｓ２，Ｓ３）。

【００２２】図４（ａ）は、このラベルのフォーマットを示している。すなわち、各プログラムモジュールの分岐節に書き込まれるラベルは、クラス名Ｌ１と、メンバ

関数名（メソッド名）Ｌ２と、「if, for, switch」文等の分岐文種別Ｌ３と、着目する関数内のインデックスとなる関数内通番（ラベル番号という）Ｌ４と、「then, else, case」節等の節種別Ｌ５とから構成される。このラベルの情報により、分岐文の位置、すなわち、ラベルを与えるプログラム上の位置（処理ルート）が明確化される。つまり、クラス名、メンバ関数名は、着目する関数で使用されているものがそのまま使用されるため、この表記によって、ラベルを見るだけで「どのクラスに属する、どのメンバ関数の、どの分岐点の、どの分岐節に付けられたラベルであるか」が直ちに分かるようになっている。

【００２３】図４（ｂ）は、プログラムモジュールにおいて、分岐点Ｐから派生した分岐節Ｂ１，Ｂ２にラベルが実際に書き込まれた例を示している。この図において、①，②に示す「printf」文の行がラベルである。このようなラベルをプログラムモジュールに埋め込むことにより、①では、「Procという名前のクラスに属する、proc1というメンバ関数内のif文中の通番１０００番のthen節における処理である」という情報が、デバッガを起動して単体試験を実行する際、printf文によって標準出力に出力される。

【００２４】処理分岐点ラベル付与部２において実行される、このようなプログラムモジュールに対するラベルの書き込み処理の内容を図５に示している。このラベル書き込み処理において、まず、ラベル番号の初期値（例えば１０００番）が設定される（Ｓ２０）。次に、構文解析部２ａでは、プログラムモジュールを１行単位で読み込み（Ｓ２１）、読み込んだ文字列中に分岐節があるか否かが判断される（Ｓ２２）。分岐節がない場合（Ｓ２２；ＮＯ）、ステップＳ２６に移行する。一方、読み込んだ文字列中に分岐節がある場合、ラベル生成・書込部２ｂは、上述のラベルを生成し（Ｓ２３）、このラベルをプログラムモジュール中の分岐節に書き込む処理を行う（Ｓ２４）。そして、ラベル番号の値を所定数だけインクリメントして（Ｓ２５）、ステップＳ２１で読み込んだソースモジュール中の行が最終行であるか否かが判断される（Ｓ２６）。最終行である場合（Ｓ２６；ＹＥＳ）、プログラムモジュールに対するラベル付与処理を終了する。一方、最終行ではない場合（Ｓ２６；ＮＯ）、次の行へポインタを移して（Ｓ２７）、再度ステップＳ２１以下の処理が実行される。

【００２５】再び、図３の説明に戻る。このようにして、母体プログラムモジュールおよび改版プログラムモジュールに対してラベルが埋め込まれ（Ｓ２，Ｓ３）、分岐点ラベル付プログラムモジュール記憶部３に記憶されると、プログラムモジュール比較照合部４ａは、これらのモジュールを読み出して、まず、ラベル付き改版プログラムモジュールの最初の分岐節へポインタを移動する（Ｓ４）。そして、母体ソースモジュールに該当する分岐節があるか否かが判断される（Ｓ５）。母体ソースモジュールに該当する分岐節がない場合（Ｓ５；ＹＥＳ）、当該分岐節は新たに追加されたものと考えられるから、試験対象ルートとすべく、改版プログラムモジュールの着目している分岐節に付与されたラベルを出力する（Ｓ６）。一方、母体ソースモジュールに該当する分岐節がある場合（Ｓ５；ＮＯ）、次のようにして、双方モジュールの分岐節同士で、当該分岐節における処理の内容が比較照合される（Ｓ７）。

【００２６】図６は、この分岐節同士の処理内容を比較照合する処理を示すフローチャートである。まず、改版ソースモジュールの分岐節の処理内容と、母体ソースモジュールにおける該当する分岐節の処理内容とを比較対照しつつ、両者の差異が解析される（Ｓ３０）。そして、双方の分岐節において、ラベルが異なるか否かが判断される（Ｓ３１）。ラベルが同じ場合（Ｓ３１；ＮＯ）、分岐節中の処理内容が異なるか否かが判断される（Ｓ３２）。処理内容が異なる場合（Ｓ３２；ＹＥＳ）、分岐条件は同一であるが、分岐節中の処理内容が変更されたと考えられるので、試験対象ルートとすべく、改版プログラムモジュールのラベルを書き換えてから、書き換えられたラベルを出力する（Ｓ３３）。一方、処理内容が同一の場合（Ｓ３２；ＮＯ）には、着目している分岐節部分は、改版前後で何ら変更がないと考えられるから図６に示す処理を終了する。

【００２７】一方、ステップＳ３１でラベルが異なると判定された場合（Ｓ３１；ＹＥＳ）、上記ステップＳ３２と同様、分岐節中の処理内容が異なるか否かが判断される（Ｓ３４）。処理内容が同じ場合（Ｓ３４；ＮＯ）、改版前後で処理内容は同一でラベル番号のみ異なっていると考えられるから、改版プログラムモジュールの分岐節中のラベル番号を、母体プログラムモジュールの該当する分岐節に付与されたラベル番号に書き換える処理が実行される（Ｓ３５）。一方、分岐節中の処理内容が異なる場合（Ｓ３４；ＹＥＳ）、つまり、分岐節中のラベルも処理内容も異なる場合には、改版プログラムモジュールにおいて、変更された処理であると考えられるから、試験対象ルートとすべく、改版プログラムモジュールの分岐節中のラベルを出力する（Ｓ３６）。以上のようにして、分岐節同士の処理内容を比較照合する処理が実行される。

【００２８】再び、図３の説明に戻る。このようにステップＳ７で分岐節の内容の比較照合処理が終了すると、改版プログラムモジュールのすべての分岐節に対して、比較照合が終了したか否かが判断され（Ｓ８）、すべての分岐節に対して比較照合が終了した場合には（Ｓ８；ＹＥＳ）、一連の試験項目抽出処理が完了する。一方、すべての分岐節に対して比較照合が終了していない場合（Ｓ８；ＮＯ）、改版プログラムモジュールの次の分岐節へポインタを移動し（Ｓ９）、再びステップＳ５以下

の処理が実行される。

【００２９】以上にようにして、試験ポイント抽出部４で試験対象となるルートを示すラベルが次々と出力され、試験ポイント一覧生成・出力部５に供給されると、すべての試験対象ルートを示す一連のラベル群が適宜な形式でファイルにまとめられ、試験担当者に試験項目一覧リストとして提供される。

【００３０】３．実施形態の効果

（１）プログラムの分岐ポイントの網羅性をキーとして単体試験項目を自動抽出することにより、試験項目抽出作業において項目漏れや冗長がなくなり、試験項目抽出作業の品質が均一化されるとともに、試験項目の抽出作業の効率化を図ることができる。

（２）分岐節に付与するラベルにクラス名、メンバ関数、分岐文の種別等の情報をもたせることとしたから、試験対象項目を示すラベルを見れば、膨大な数のソースモジュールの中から直ちに試験ポイントを把握することができる。したがって、実際の単体試験において、シンボリックデバッガを立ち上げ、試験対象ルートに書き込まれたラベルの位置までカーソルをとばしてブレイクポイントを設定することにより、直ちに単体試験が可能である。

【００３１】４．変形例

本発明は、上記実施形態に限定されるものではなく、例えば以下のように種々の変形が可能である。

（１）各プログラムモジュールの各分岐節に付与されるラベルは、モジュール中に直接書き込むこととしたが、このラベルを母体プログラムモジュールと改版プログラムモジュールとでそれぞれ別のテーブルに書き込むようにしておき、このテーブル同士を比較するようにしてもよい。

（２）プログラミング言語としてＣ＋＋を対象にしたが、Ｃ，ＪＡＶＡ等種々のプログラミング言語で記述されたプログラムモジュールに対しても適用が可能である。

【００３２】

＊【発明の効果】以上詳細に説明したように、この発明によれば、試験対象のプログラムモジュールに対して、試験対象項目が自動的に抽出されるので、試験項目の漏れがなくなるとともに、試験担当者の負荷も大幅に軽減される。

【図面の簡単な説明】

【図１】本実施形態にかかる試験項目抽出装置の機能構成を示す図である。

【図２】同上試験項目抽出装置のハードウェア構成の概略を示す図である。

【図３】同上試験項目抽出装置における処理の流れを示すフローチャートである。

【図４】同上試験項目抽出装置において付与されるラベルの内容を示し、（ａ）はラベルのフォーマットを、（ｂ）はプログラムモジュールにラベルが埋め込まれた例を示している。

【図５】同上試験項目抽出装置において、プログラムモジュールに対するラベル付与処理の流れを示すフローチャートである。

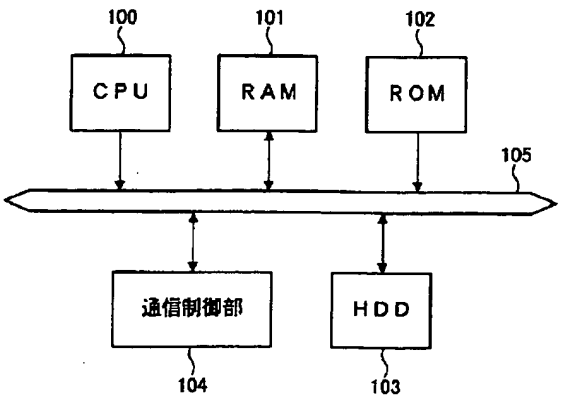
【図６】同上試験項目抽出装置において、分岐節の内容を比較照合する処理の流れを示すフローチャートである。

【図７】従来の試験項目抽出方法のプロセスを単体プログラム内の処理分岐モデルによって示したものである。

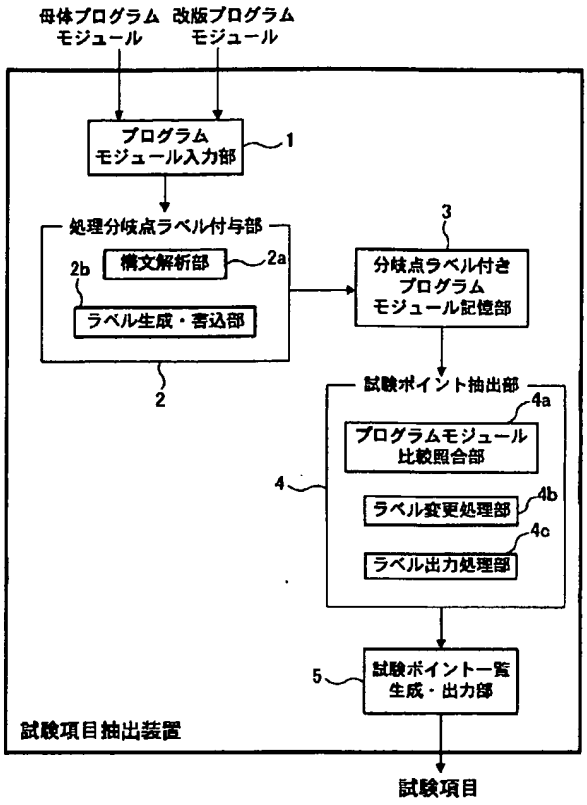
【符号の説明】

1　プログラムモジュール入力部
2　処理分岐点ラベル付与部
2ａ　構文解析部
2ｂ　ラベル生成・書込部
3　分岐点ラベル付プログラムモジュール記憶部
4　試験ポイント抽出部
4ａ　プログラムモジュール比較照合部
4ｂ　ラベル変更処理部
4ｃ　ラベル出力処理部
5　試験ポイント一覧生成・出力部

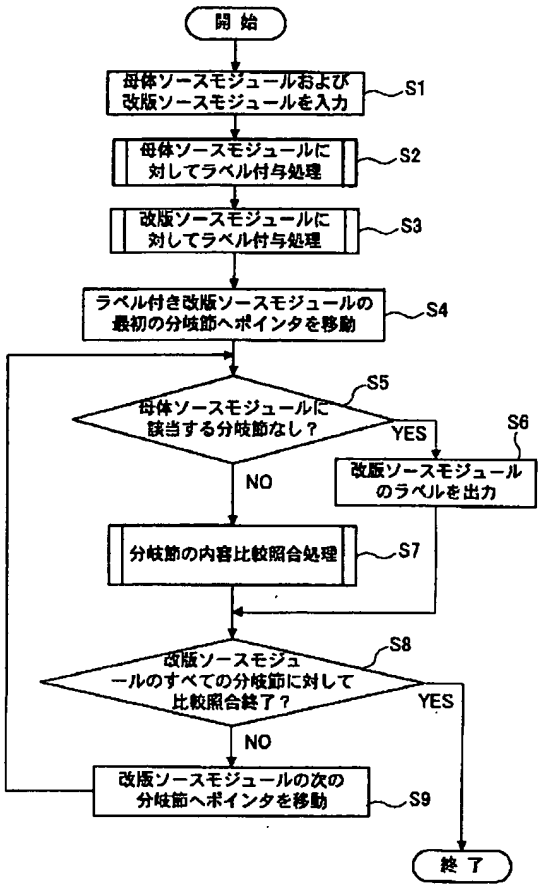
【図２】



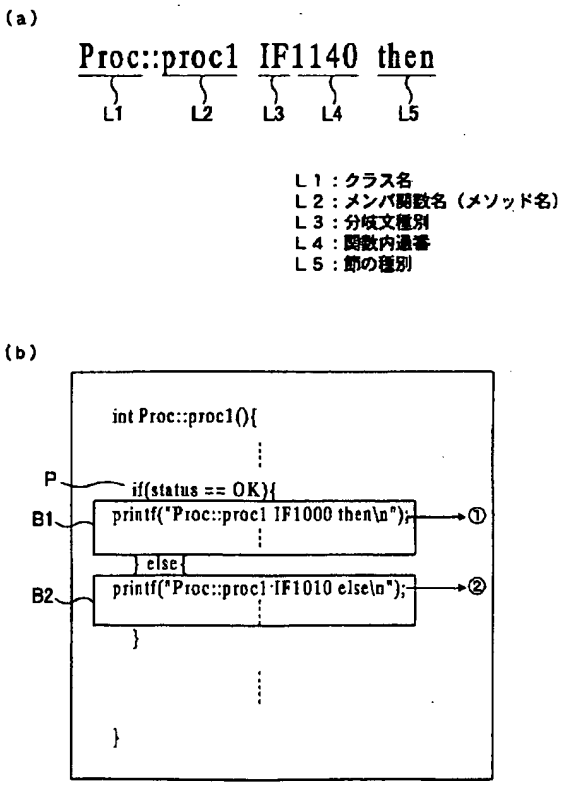
【図１】



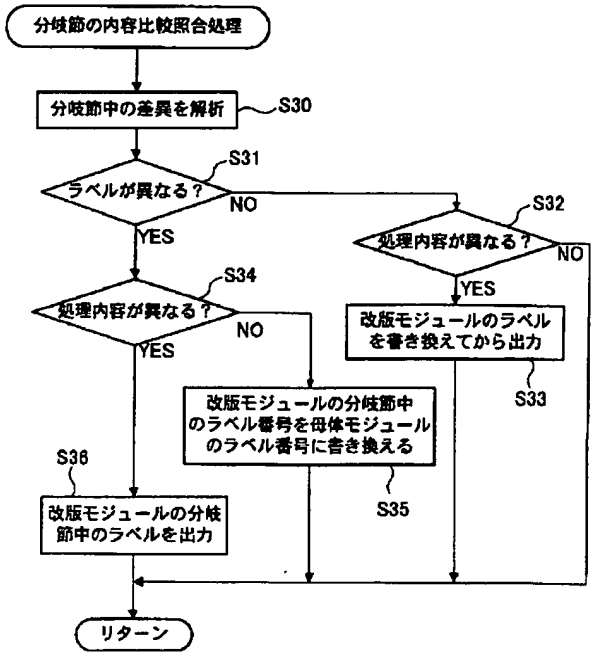
【図３】



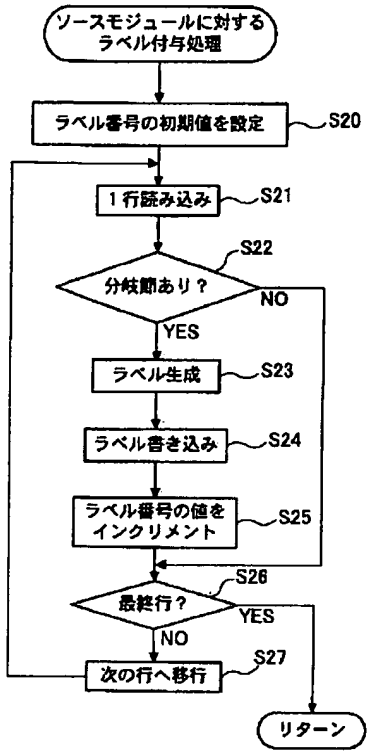
【図４】



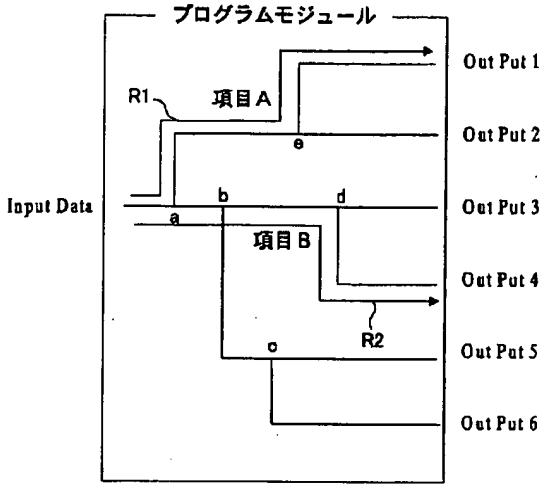
【図６】



【図５】



【図７】



フロントページの続き

(72)発明者　松谷　忠行
東京都港区港南一丁目９番１号　エヌ・ティ・ティ・コミュニケーションウェア株式会社内

(72)発明者　服部　信隆
東京都港区港南一丁目９番１号　エヌ・ティ・ティ・コミュニケーションウェア株式会社内

(72)発明者　安藤　金吾
東京都港区港南一丁目９番１号　エヌ・ティ・ティ・コミュニケーションウェア株式会社内

Ｆターム(参考)　5B042 GA08 HH10 HH17 HH44